Statistical Aspects of Quantum Computing

Yazhen Wang

Department of Statistics

University of Wisconsin-Madison

http://www.stat.wisc.edu/~yzwang

Near-term Applications of Quantum Computing

Fermilab, December 6-7, 2017

Outline

- Statistical learning with quantum annealing
- Statistical analysis of quantum computing data

Statistics and Optimization

MLE/M-estimation, Non-parametric smoothing, $\cdots$

- Stochastic optimization problem: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; X_i)$

- Minimization solution gives an estimator or a classifier.
  Examples : $\ell(\theta; X_i) = \log pdf$; residual square sum / loss + penalty

Statistics and Optimization

MLE/M-estimation, Non-parametric smoothing, $\cdots$

- Stochastic optimization problem: $\min_{\theta} \mathcal{L}(\theta; \mathbf{X}_n) = \dfrac{1}{n}\sum_{i=1}^{n} \ell(\theta; X_i)$

• Minimization solution gives an estimator or a classifier.
Examples : $\ell(\theta; X_i) = \log pdf$; residual square sum / loss + penalty

Take $g(\theta) = E[\mathcal{L}(\theta; \mathbf{X}_n)] = E[\ell(\theta; X_1)]$

- Optimization problem: $\min_{\theta} g(\theta)$

• Minimization solution defines a true parameter value.

Statistics and Optimization

MLE/M-estimation, Non-parametric smoothing, $\cdots$

- Stochastic optimization problem: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; X_i)$

• Minimization solution gives an estimator or a classifier.
Examples : $\ell(\theta; X_i) = \log pdf$; residual square sum / loss + penalty

Take $g(\theta) = E[\mathcal{L}(\theta; \mathbf{X}_n)] = E[\ell(\theta; X_1)]$

- Optimization problem: $\min_\theta g(\theta)$

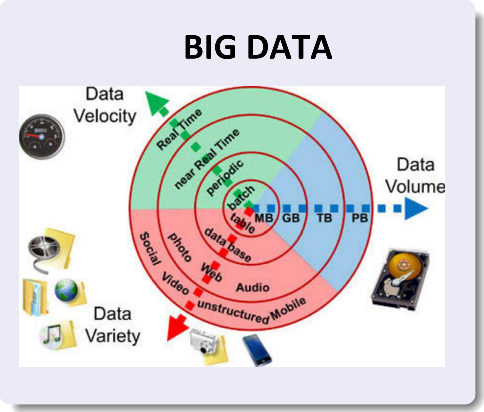
• Minimization solution defines a true parameter value.

Goals: Use data $\mathbf{X}_n$ to do the following

(i) Evaluate estimators/classifiers (minimization solutions) **Computing**
(ii) Statistical study of estimators/classifiers – **Inference**

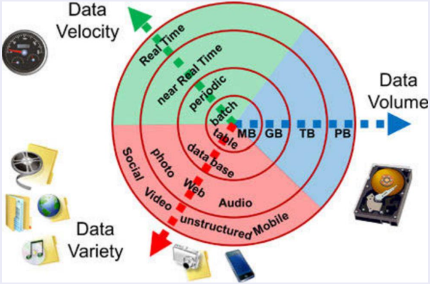
Computer Power Demand

Computer Power Demand

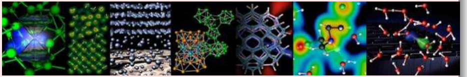


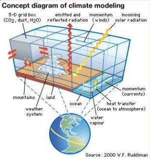
Computer Power Demand

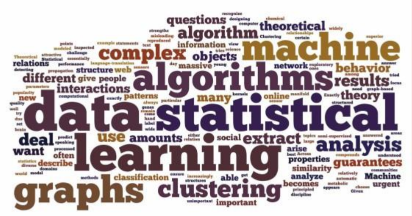
BIG DATA



Scientific Studies and Computational Applications







Learning examples

Machine learning and compressed sensing

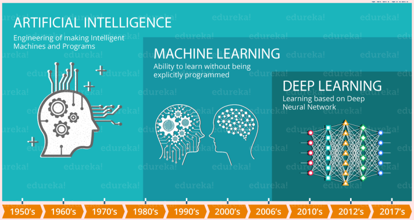
• Matrix completion, matrix factorization, tensor decomposition, phase retrieval, neural network.

Learning examples

Machine learning and compressed sensing

• Matrix completion, matrix factorization, tensor decomposition, phase retrieval, neural network.

History

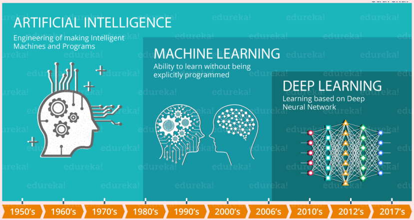


Learning examples

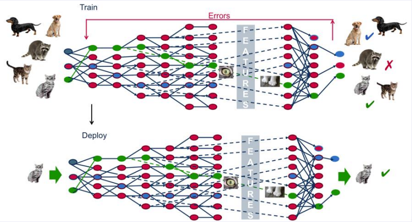
Machine learning and compressed sensing

• Matrix completion, matrix factorization, tensor decomposition, phase retrieval, neural network.

History



Dog vs cat



Learning examples

Machine learning and compressed sensing

• Matrix completion, matrix factorization, tensor decomposition, phase retrieval, neural network.

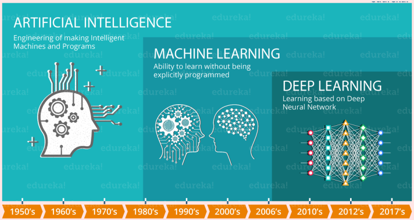
Neural network: Layers in a chain structure

Each layer is a function of the layer preceded it.

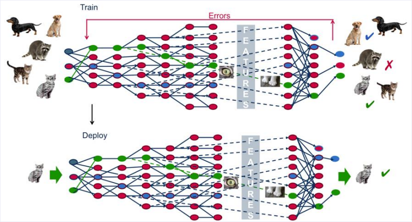
Layer j: $h_j = g_j(a_j h_{j-1} + b_j)$, $(a_j, b_j) =$ weights,

$g_j =$ activation function (sigmoid, softmax or rectifier)

History



Dog vs cat



Gradient Descent Alorithms: Solve $\min_\theta g(\theta)$

Gradient descent algorithm

- Start at initial value x_0,
  $x_k = x_{k-1} - \delta \nabla g(x_{k-1}), \;\; \delta =$ learning rate, $\nabla =$ derivative operator

Gradient Descent Alorithms: Solve $\min_\theta g(\theta)$

Gradient descent algorithm

- Start at initial value x_0,
  $x_k = x_{k-1} - \delta \nabla g(x_{k-1}), \quad \delta = \text{learning rate}, \ \nabla = \text{derivative operator}$

Accelerated Gradient descent algorithm (Nesterov)

- Start at initial values x_0 and $y_0 = x_0$,

$$x_k = y_{k-1} - \delta \nabla g(y_{k-1}), \qquad y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Gradient Descent Alorithms: Solve $\min_\theta g(\theta)$

Gradient descent algorithm

- Start at initial value x_0,
  $x_k = x_{k-1} - \delta \nabla g(x_{k-1})$, $\delta =$ learning rate, $\nabla =$ derivative operator

Continuous curve X_t to approximate discrete $\{x_k : k \geq 0\}$

Differential equation: $\dot{X}_t + \nabla g(X_t) = 0$, $\dot{X}_t =$ derivative $= \dfrac{dX_t}{dt}$

Accelerated Gradient descent algorithm (Nesterov)

- Start at initial values x_0 and $y_0 = x_0$,

$$x_k = y_{k-1} - \delta \nabla g(y_{k-1}), \qquad y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Gradient Descent Alorithms: Solve $\min_\theta g(\theta)$

Gradient descent algorithm

- Start at initial value x_0,
  $x_k = x_{k-1} - \delta \nabla g(x_{k-1})$, $\delta =$ learning rate, $\nabla =$ derivative operator

Continuous curve X_t to approximate discrete $\{x_k : k \geq 0\}$

Differential equation: $\dot{X}_t + \nabla g(X_t) = 0$, $\dot{X}_t =$ derivative $= \dfrac{dX_t}{dt}$

Accelerated Gradient descent algorithm (Nesterov)

- Start at initial values x_0 and $y_0 = x_0$,

$$x_k = y_{k-1} - \delta \nabla g(y_{k-1}), \qquad y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Continuous curve X_t to approximate discrete $\{x_k : k \geq 0\}$

Differential equation: $\ddot{X}_t + \dfrac{3}{t}\dot{X}_t + \nabla g(X_t) = 0$, $\ddot{X}_t = \dfrac{d^2 X_t}{dt^2}$

Gradient Descent Alorithms: Solve $\min_\theta g(\theta)$

Gradient descent algorithm

- Start at initial value x_0,
  $x_k = x_{k-1} - \delta \nabla g(x_{k-1})$, $\delta =$ learning rate, $\nabla =$ derivative operator

Continuous curve X_t to approximate discrete $\{x_k : k \geq 0\}$

Differential equation: $\dot{X}_t + \nabla g(X_t) = 0$, $\dot{X}_t =$ derivative $= \frac{dX_t}{dt}$

Convergence to the minimization solution at rate= $1/k$ or $1/t$ $(\uparrow)$

as $t, k \to \infty$. For the ccelerated case: Rate = $1/k^2$ or $1/t^2 (\downarrow)$

Accelerated Gradient descent algorithm (Nesterov)

- Start at initial values x_0 and $y_0 = x_0$,

$$x_k = y_{k-1} - \delta \nabla g(y_{k-1}), \qquad y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Continuous curve X_t to approximate discrete $\{x_k : k \geq 0\}$

Differential equation: $\ddot{X}_t + \frac{3}{t}\dot{X}_t + \nabla g(X_t) = 0$, $\ddot{X}_t = \frac{d^2 X_t}{dt^2}$

Stochastic Gradient Descent

Stochastic optimization: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n)$, $\mathbf{X}_n = (X_1, \cdots, X_n)$

- Gradient descent algorithm to compute x_k iteratively

$$x_k = x_{k-1} - \delta \nabla \mathcal{L}(x_{k-1}; \mathbf{X}_n), \quad \nabla \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\theta; X_i)$$

Stochastic Gradient Descent

Stochastic optimization: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n)$, $\mathbf{X}_n = (X_1, \cdots, X_n)$

- Gradient descent algorithm to compute x_k iteratively

$$x_k = x_{k-1} - \delta \nabla \mathcal{L}(x_{k-1}; \mathbf{X}_n), \quad \nabla \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\theta; X_i)$$

BigData: expensive to evaluate all $\nabla \ell(\theta; X_i)$ at each iteration

- Replace $\nabla \mathcal{L}(\theta; \mathbf{X}_n)$ by

$$\nabla \hat{\mathcal{L}}^m(\theta; \mathbf{X}^*_m) = \frac{1}{m} \sum_{j=1}^{m} \nabla \ell(\theta; X^*_j), \qquad m \ll n$$

$\mathbf{X}^*_m = (X^*_1, \cdots, X^*_m)$= subsample of $\mathbf{X}_n$ (minibatch or bootstrap sample).

Stochastic Gradient Descent

Stochastic optimization: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n)$, $\mathbf{X}_n = (X_1, \cdots, X_n)$

- Gradient descent algorithm to compute x_k iteratively

$$x_k = x_{k-1} - \delta \nabla \mathcal{L}(x_{k-1}; \mathbf{X}_n), \quad \nabla \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\theta; X_i)$$

BigData: expensive to evaluate all $\nabla \ell(\theta; X_i)$ at each iteration

- Replace $\nabla \mathcal{L}(\theta; \mathbf{X}_n)$ by

$$\nabla \hat{\mathcal{L}}^m(\theta; \mathbf{X}_m^*) = \frac{1}{m} \sum_{j=1}^{m} \nabla \ell(\theta; X_j^*), \qquad m \ll n$$

$\mathbf{X}_m^* = (X_1^*, \cdots, X_m^*)$= subsample of $\mathbf{X}_n$ (minibatch or bootstrap sample).

Stochastic gradient descent algorithm

$$x_k^* = x_{k-1}^* - \delta \nabla \hat{\mathcal{L}}^m(x_{k-1}^*; \mathbf{X}_m^*)$$

Stochastic Gradient Descent

Stochastic optimization: $\min_\theta \mathcal{L}(\theta; \mathbf{X}_n)$, $\mathbf{X}_n = (X_1, \cdots, X_n)$

- Gradient descent algorithm to compute x_k iteratively

$$x_k = x_{k-1} - \delta \nabla \mathcal{L}(x_{k-1}; \mathbf{X}_n), \quad \nabla \mathcal{L}(\theta; \mathbf{X}_n) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\theta; X_i)$$

BigData: expensive to evaluate all $\nabla \ell(\theta; X_i)$ at each iteration

- Replace $\nabla \mathcal{L}(\theta; \mathbf{X}_n)$ by

$$\nabla \hat{\mathcal{L}}^m(\theta; \mathbf{X}_m^*) = \frac{1}{m} \sum_{j=1}^{m} \nabla \ell(\theta; X_j^*), \qquad m \ll n$$

$\mathbf{X}_m^* = (X_1^*, \cdots, X_m^*)$= subsample of $\mathbf{X}_n$ (minibatch or bootstrap sample).

Stochastic gradient descent algorithm

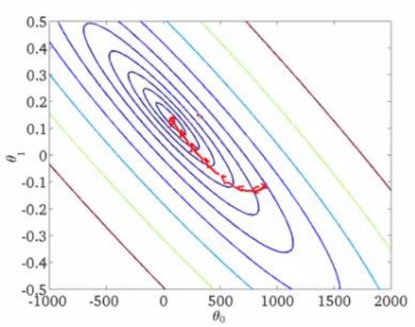
$$x_k^* = x_{k-1}^* - \delta \nabla \hat{\mathcal{L}}^m(x_{k-1}^*; \mathbf{X}_m^*)$$

Continuous curve X_t^* to approximate discrete $\{x_k^* : k \geq 0\}$

X_t^* obeys stochastic differential equation.

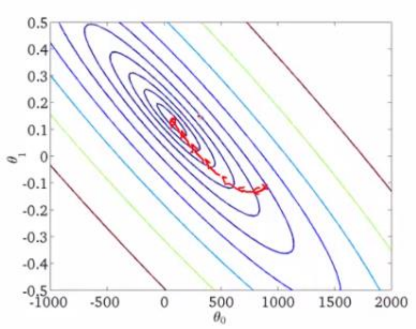
Gradient Descent vs Stochastic Gradient Descent

Gradient Descent

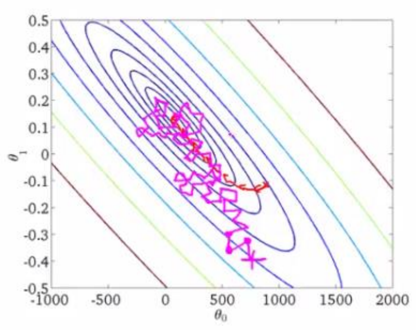


Gradient Descent vs Stochastic Gradient Descent

Gradient Descent



Stochastic gradient descent



Statistical Analysis of Gradient Descent (Wang, 2017)

Continuous curve model

Stochastic differential equation:

$$dX_t^* + \nabla g(X_t^*)dt + \sigma(X_t^*)dW_t = 0$$

W_t = Brownian motion

For the accelerated case:

2nd order stochastic differential equation

Statistical Analysis of Gradient Descent (Wang, 2017)

Continuous curve model

Stochastic differential equation:
$dX_t^* + \nabla g(X_t^*)dt + \sigma(X_t^*)dW_t = 0$
W_t = Brownian motion
For the accelerated case:
2nd order stochastic differential equation

and their asymptotic distribution

as $m, n \to \infty$ via stochastic differential equations

Statistical Analysis of Gradient Descent (Wang, 2017)

Continuous curve model

Stochastic differential equation:

$$dX_t^* + \nabla g(X_t^*)dt + \boldsymbol{\sigma}(X_t^*)dW_t = 0$$

W_t = Brownian motion

For the accelerated case:

2nd order stochastic differential equation

and their asymptotic distribution

as $m, n \to \infty$ via stochastic differential equations

Example $X_i = (U_i, V_i)$, $i = 1, \cdots, n = 10000$

$$V_i = U_i\theta + \varepsilon_i, \quad U_i \sim i.i.d.\text{bivariate}N(0, \Sigma), \varepsilon_i \sim i.i.d.N(0, \tau^2)$$

$\ell(\theta; X_i) = (V_i - U_i\theta)^2$, $m = 200$, true $\theta = (0, 0)$.

Statistical Analysis of Gradient Descent (Wang, 2017)

Continuous curve model

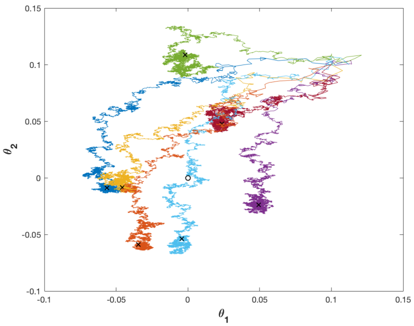
Stochastic differential equation:

$$dX_t^* + \nabla g(X_t^*)dt + \boldsymbol{\sigma}(X_t^*)dW_t = 0$$

W_t = Brownian motion

For the accelerated case:

2nd order stochastic differential equation



and their asymptotic distribution

as $m, n \to \infty$ via stochastic differential equations

Example $X_i = (U_i, V_i)$, $i = 1, \cdots, n = 10000$

$$V_i = U_i\theta + \varepsilon_i, \quad U_i \sim i.i.d.\text{bivariate}N(0, \Sigma), \varepsilon_i \sim i.i.d.N(0, \tau^2)$$

$\ell(\theta; X_i) = (V_i - U_i\theta)^2$, $m = 200$, true $\theta = (0, 0)$.

Deep Learning

Boltzmann Machine (BM) on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- $$P(\mathbf{s}) = \frac{\exp[-E(\mathbf{s})]}{Z}, \quad Z = \sum_{\mathbf{s}} \exp[-E(\mathbf{s})]$$
- Energy
$$E(\mathbf{s}) = -\sum_{(i,j)\in\mathcal{E}} W_{ij} s_i s_j - \sum_{i\in\mathcal{V}} b_i s_i, \quad \mathbf{s} = (s_1, \cdots, s_{|\mathcal{V}|}) \in \{-1, 1\}^{|\mathcal{V}|}$$

Deep Learning

Boltzmann Machine (BM) on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- $$P(\mathbf{s}) = \frac{\exp[-E(\mathbf{s})]}{Z}, \quad Z = \sum_{\mathbf{s}} \exp[-E(\mathbf{s})]$$
- Energy
$$E(\mathbf{s}) = -\sum_{(i,j)\in\mathcal{E}} W_{ij} s_i s_j - \sum_{i\in\mathcal{V}} b_i s_i, \quad \mathbf{s} = (s_1, \cdots, s_{|\mathcal{V}|}) \in \{-1, 1\}^{|\mathcal{V}|}$$

Take $\mathbf{s} = (\mathbf{v}, \boldsymbol{h})$

$\mathbf{v} = (\mathbf{v_1}, \cdots, \mathbf{v_n})$: visible nodes (observed variables)

$\boldsymbol{h} = (h_1, \cdots, h_m)$: hidden nodes (latent variables).

Boltzmann distribution models data $\mathbf{v}$:

$$P(\mathbf{v}) = \sum_{\boldsymbol{h}} P(\mathbf{v}, \boldsymbol{h})$$

Deep Learning

Boltzmann Machine (BM) on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- $$P(\mathbf{s}) = \frac{\exp[-E(\mathbf{s})]}{Z}, \quad Z = \sum_{\mathbf{s}} \exp[-E(\mathbf{s})]$$
- Energy
$$E(\mathbf{s}) = -\sum_{(i,j)\in\mathcal{E}} W_{ij} s_i s_j - \sum_{i\in\mathcal{V}} b_i s_i, \quad \mathbf{s} = (s_1, \cdots, s_{|\mathcal{V}|}) \in \{-1, 1\}^{|\mathcal{V}|}$$

Take $\mathbf{s} = (\mathbf{v}, \boldsymbol{h})$

$\mathbf{v} = (\mathbf{v_1}, \cdots, \mathbf{v_n})$: visible nodes (observed variables)
$\boldsymbol{h} = (h_1, \cdots, h_m)$: hidden nodes (latent variables).
Boltzmann distribution models data $\mathbf{v}$:

$$P(\mathbf{v}) = \sum_{\boldsymbol{h}} P(\mathbf{v}, \boldsymbol{h})$$

Learning

Use training data $\mathbf{v}$ to learn model parameters W_{ij} & b_i.

Restricted Boltzmann Machine (RBM)

Bipartite undirected graph $\mathcal{G}$

Connections between hidden layer
and visible layer
but not within each layer

Restricted Boltzmann Machine (RBM)

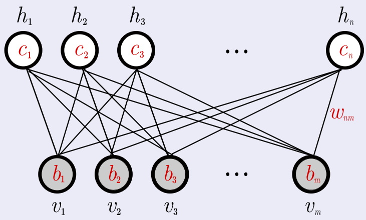
Bipartite undirected graph $\mathcal{G}$

Connections between hidden layer and visible layer
but not within each layer

Model

Variables in visible layer:
$\mathbf{v} = (v_1, \cdots, v_n)$,
Variables in hidden layer:
$\boldsymbol{h} = (h_1, \cdots, h_m)$

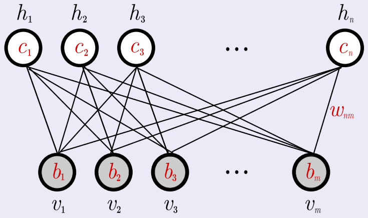
$$P(\mathbf{v}, \boldsymbol{h}) = \exp\{-E(\mathbf{v}, \boldsymbol{h})\}/Z$$



Restricted Boltzmann Machine (RBM)

Bipartite undirected graph $\mathcal{G}$

Connections between hidden layer and visible layer
but not within each layer



Model

Variables in visible layer:
$\mathbf{v} = (v_1, \cdots, v_n)$,
Variables in hidden layer:
$\boldsymbol{h} = (h_1, \cdots, h_m)$

$$P(\mathbf{v}, \boldsymbol{h}) = \exp\{-E(\mathbf{v}, \boldsymbol{h})\}/Z$$

$$E(\mathbf{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j - \sum_{i=1}^{n} b_i v_i - \sum_{j=1}^{m} c_j h_j$$

Deep Neural Network: Restricted Boltzmann Machine

Deep Neural Network: Restricted Boltzmann Machine

Conditional independence within each layer given the others

$$P(\boldsymbol{h}|\mathbf{v}) = \prod_{j=1}^{m} P(h_j|\mathbf{v}), \quad P(\mathbf{v}|\boldsymbol{h}) = \prod_{i=1}^{n} P(v_i|\boldsymbol{h})$$

Deep Neural Network: Restricted Boltzmann Machine

Conditional independence within each layer given the others

$$P(\boldsymbol{h}|\mathbf{v}) = \prod_{j=1}^{m} P(h_j|\mathbf{v}), \quad P(\mathbf{v}|\boldsymbol{h}) = \prod_{i=1}^{n} P(v_i|\boldsymbol{h})$$

Sigmoid activation function for forward and backward conditional probabilities: sigmoid$(x) = 1/[1 + e^{-x}]$

$$P(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(\sum_{i=1}^{n} w_{ij} v_i + c_j\right)$$

$$P(v_i = 1|\boldsymbol{h}) = \text{sigmoid}\left(\sum_{j=1}^{n} w_{ij} h_j + b_i\right)$$

Deep Learning

Gradient ascent/descent to compute model parameters w_{ij}, b_i and c_j.

Deep Learning

Gradient ascent/descent to compute model parameters w_{ij}, b_i and c_j.

Parameter updates with learning rate η

$$w_{ij}^{(t+1)} = w_{ij}^{t} + \eta \frac{\partial \log P}{\partial w_{ij}}$$

$$b_i^{(t+1)} = b_i^{t} + \eta \frac{\partial \log P}{\partial b_i}, \quad c_j^{(t+1)} = c_j^{t} + \eta \frac{\partial \log P}{\partial c_j}$$

Deep Learning

Gradient ascent/descent to compute model parameters w_{ij}, b_i and c_j.

Gradient

$$\frac{\partial \log P}{\partial w_{ij}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}$$

$$\frac{\partial \log P}{\partial b_i} = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}, \quad \frac{\partial \log P}{\partial c_j} = \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}$$

- $\langle v_i h_j \rangle_{\text{data}}$: the clamped expectation with **v** fixed

Bottleneck : $$\langle v_i h_j \rangle_{\text{model}} = \sum_{\mathbf{v}, \boldsymbol{h}} v_i h_j P(\mathbf{v}, \boldsymbol{h})$$

Parameter updates with learning rate η

$$w_{ij}^{(t+1)} = w_{ij}^{t} + \eta \frac{\partial \log P}{\partial w_{ij}}$$

$$b_i^{(t+1)} = b_i^{t} + \eta \frac{\partial \log P}{\partial b_i}, \quad c_j^{(t+1)} = c_j^{t} + \eta \frac{\partial \log P}{\partial c_j}$$

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings algorithm/Gibbs sampler

Sample from Boltzmann distribution

$$P(\mathbf{s}) = \frac{\exp[-H_{Ising}(\mathbf{s})/T]}{Z_T}, Z_T = \sum_{\mathbf{s}} \exp\left[-\frac{H_{Ising}(\mathbf{s})}{T}\right], T = \text{temperature}$$

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings algorithm/Gibbs sampler

Sample from Boltzmann distribution

$$P(\mathbf{s}) = \frac{\exp[-H_{Ising}(\mathbf{s})/T]}{Z_T}, Z_T = \sum_{\mathbf{s}} \exp\left[-\frac{H_{Ising}(\mathbf{s})}{T}\right], T = \text{temperature}$$

Simulated annealing: Thermal Fluctuation

Slowly lower the temperature to reduce the escape probability of trapping in local minima,

$$\text{Annealing schedule}: T_i \propto \frac{1}{i+1} \text{ or } \frac{1}{\log(i+1)}$$

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings algorithm/Gibbs sampler

Sample from Boltzmann distribution

$$P(\mathbf{s}) = \frac{\exp[-H_{Ising}(\mathbf{s})/T]}{Z_T}, Z_T = \sum_{\mathbf{s}} \exp\left[-\frac{H_{Ising}(\mathbf{s})}{T}\right], T = \text{temperature}$$

Simulated annealing: Thermal Fluctuation

Slowly lower the temperature to reduce the escape probability of trapping in local minima,

$$\text{Annealing schedule}: T_i \propto \frac{1}{i+1} \text{ or } \frac{1}{\log(i+1)}$$

BigData

Issues: not easy for parallel computing; very hard to scale-up!

Quantum Annealing (QA): Basic Idea

Classical optimization: Min$\{H_{Ising}(\mathbf{s}) : \mathbf{s} \in \{-1, 1\}^N\}$

Quantum Annealing (QA): Basic Idea

Classical optimization: $\text{Min}\{H_{Ising}(\mathbf{s}) : \mathbf{s} \in \{-1, 1\}^N\}$

Find a target quantum system with Hamiltonian $H(1)$ whose energies match $H_{Ising}(\mathbf{s})$: $H(1) = \text{diag}\{H_{Ising}(\mathbf{s}_1,) \cdots, H_{Ising}(\mathbf{s}_{2^N})\}$.

Quantum Annealing (QA): Basic Idea

Classical optimization: $\mathrm{Min}\{H_{Ising}(\mathbf{s}) : \mathbf{s} \in \{-1, 1\}^N\}$

Find a target quantum system with Hamiltonian $H(1)$ whose

energies match $H_{Ising}(\mathbf{s})$: $H(1) = \mathrm{diag}\{H_{Ising}(\mathbf{s}_1,) \cdots, H_{Ising}(\mathbf{s}_{2^N})\}$.

Create an initial quantum system with Hamiltonian $H(0)$

whose lowest energy state is known and easy to prepare.

Quantum Annealing (QA): Basic Idea

Classical optimization: Min$\{H_{Ising}(\mathbf{s}) : \mathbf{s} \in \{-1, 1\}^N\}$

Find a target quantum system with Hamiltonian $H(1)$ whose

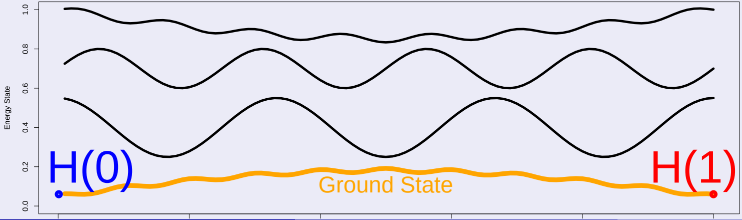
energies match $H_{Ising}(\mathbf{s})$: $H(1) = \text{diag}\{H_{Ising}(\mathbf{s}_1,) \cdots, H_{Ising}(\mathbf{s}_{2^N})\}$.

Create an initial quantum system with Hamiltonian $H(0)$

whose lowest energy state is known and easy to prepare.

QA: Engineer $H(0)$ in its lowest energy state and gradually move

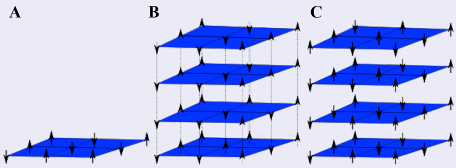
$H(0) \longrightarrow H(1)$



Simulated Quantum Annealing (SQA)

Spin glass in transverse field

$$H = \mathbf{A(t)H_X} + \mathbf{B(t)H_{Ising}}, \quad \text{two parts non-commuting}$$



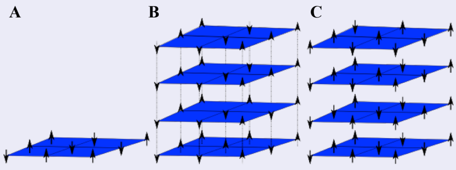
Simulated Quantum Annealing (SQA)

Spin glass in transverse field

$$H = \mathbf{A(t)H_X} + \mathbf{B(t)H_{Ising}}, \quad \text{two parts non-commuting}$$

Path integral representation via Suzuki-Trotter expansion

$H \approx H_{2+1} =$ classical (2+1)-dimensional anisotropic Ising system



Simulated Quantum Annealing (SQA)

Spin glass in transverse field

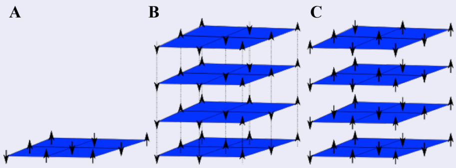
$$H = \mathbf{A(t)H_X} + \mathbf{B(t)H_{Ising}}, \quad \text{two parts non-commuting}$$

Path integral representation via Suzuki-Trotter expansion

$H \approx H_{2+1} =$ classical (2+1)-dimensional anisotropic Ising system

$(2+1)$-dimensional system

Two directions: along the original 2-dimensional direction spins have Chimera graph couplings, and along the extra (imaginary-time) direction spins have uniform couplings



Simulated Quantum Annealing (SQA)

Spin glass in transverse field

$$H = \mathbf{A(t)H_X} + \mathbf{B(t)H_{Ising}},$$ two parts non-commuting

Path integral representation via Suzuki-Trotter expansion

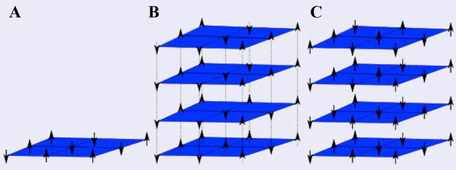
$H \approx H_{2+1}$ = classical (2+1)-dimensional anisotropic Ising system

$(2+1)$-dimensional system

Two directions: along the original 2-dimensional direction spins have Chimera graph couplings, and along the extra (imaginary-time) direction spins have uniform couplings

Quantum Monte Carlo

H_{2+1}: a collection of 2-dimensional classical Ising systems, that can be simulated by MCMC with moves in both directions



SSSV Annealing Model

Magnet i points in direction with angle θ_i w.r.t. $\vec{z}$-axis in the xz plane, an external magnetic field with intensity $A(t)$ pointing in the $\vec{x}$-axis,

Hamiltonian, J_{ij} = coupling of magnets θ_i and θ_j,

$$H(t) = -A(t)\sum_{i=1}^{N}\sin\theta_i - B(t)\sum_{1\le i<j\le N} J_{ij}\cos\theta_i\cos\theta_j$$

SSSV Annealing Model

Magnet i points in direction with angle θ_i w.r.t. $\vec{z}$-axis in the xz plane, an external magnetic field with intensity $A(t)$ pointing in the $\vec{x}$-axis,

Hamiltonian, J_{ij} = coupling of magnets θ_i and θ_j,

$$H(t) = -A(t)\sum_{i=1}^{N} \sin\theta_i - B(t)\sum_{1\le i<j\le N} J_{ij}\cos\theta_i\cos\theta_j$$

The model can be simulated by the Metropolis algorithm with temperature $T = 0.22$, and initial condition $\theta_i = \pi/2$

SSSV Annealing Model

Magnet i points in direction with angle θ_i w.r.t. $\vec{z}$-axis in the xz plane, an external magnetic field with intensity $A(t)$ pointing in the $\vec{x}$-axis,

Hamiltonian, J_{ij} = coupling of magnets θ_i and θ_j,

$$H(t) = -A(t)\sum_{i=1}^{N}\sin\theta_i - B(t)\sum_{1\le i<j\le N} J_{ij}\cos\theta_i\cos\theta_j$$

The model can be simulated by the Metropolis algorithm with temperature $T = 0.22$, and initial condition $\theta_i = \pi/2$

Interpretation: angle θ_i as state $|\uparrow\rangle$(=+1) or state $|\downarrow\rangle$(= -1) according to the sign of $\cos(\theta_i)$ (its projection on $\vec{z}$ direction).

SSSV Annealing Model

Magnet i points in direction with angle θ_i w.r.t. $\vec{z}$-axis in the xz plane, an external magnetic field with intensity $A(t)$ pointing in the $\vec{x}$-axis,

Hamiltonian, J_{ij} = coupling of magnets θ_i and θ_j,

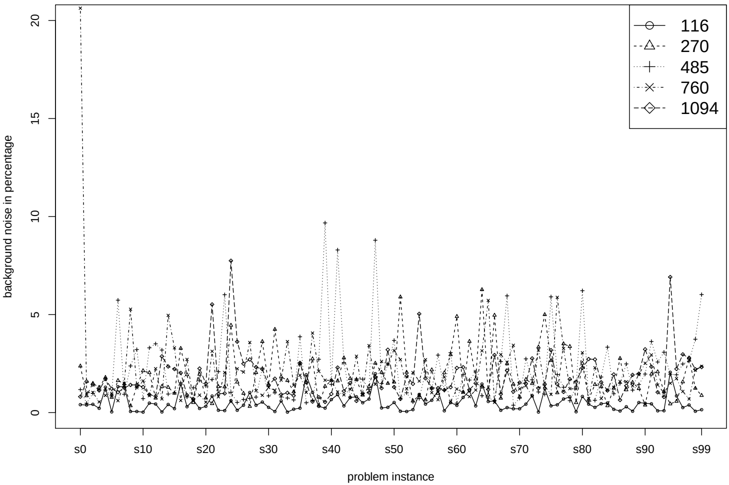
$$H(t) = -A(t)\sum_{i=1}^{N} \sin\theta_i - B(t) \sum_{1\le i<j\le N} J_{ij}\cos\theta_i\cos\theta_j$$

The model can be simulated by the Metropolis algorithm with temperature $T = 0.22$, and initial condition $\theta_i = \pi/2$

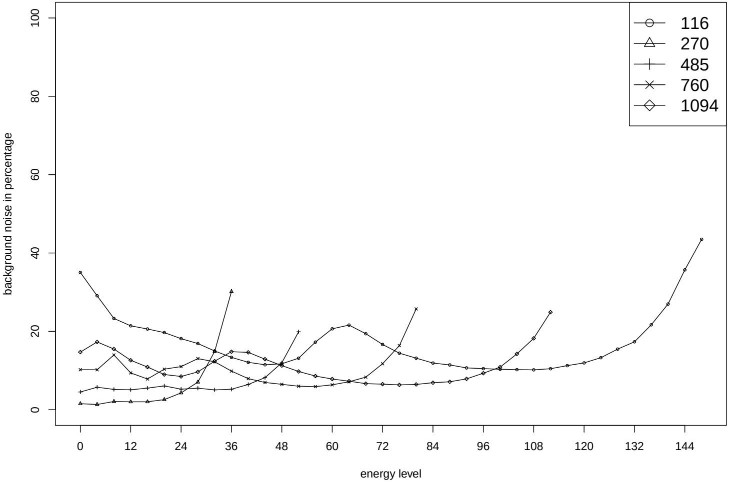
Interpretation: angle θ_i as state $|\uparrow\rangle$(=+1) or state $|\downarrow\rangle$(= -1) according to the sign of $\cos(\theta_i)$ (its projection on $\vec{z}$ direction).

Use the converted states to evaluate $H_{Ising}(\mathbf{s})$ and find its minimizer

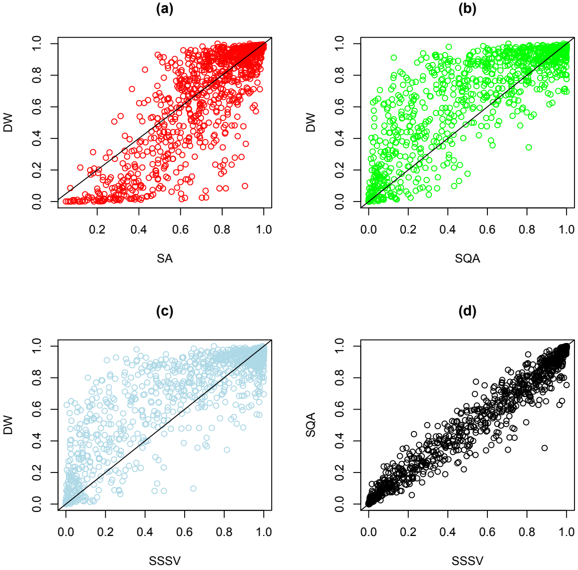
DW Signal vs Background Noise



DW Signal vs Background Noise



Correlation of Ground State Success Probability Data



Multiple Statistical Tests

For the r-th instance, repeat m times of annealing, let $\hat{p}_{0rm}$ be DW success frequency out of m repetitions and $\hat{q}_{\ell rm}$, $\ell = 1, 2, 3$, the success frequencies for SA, SQA & SSSV

Multiple Statistical Tests

For the r-th instance, repeat m times of annealing, let $\hat{p}_{0rm}$ be DW success frequency out of m repetitions and $\hat{q}_{\ell rm}$, $\ell = 1, 2, 3$, the success frequencies for SA, SQA & SSSV

$H_{0r} : p_{0r\infty} = q_{\ell r\infty}$ vs $H_{ar} : p_{0r\infty} \neq q_{\ell r\infty}$

$$T_{r\ell} = \frac{m(\hat{p}_r - \hat{q}_{\ell,r})^2}{\hat{p}_r(1 - \hat{p}_r) + \hat{q}_{\ell,r}(1 - \hat{q}_{\ell,r})}$$

Multiple Statistical Tests

For the r-th instance, repeat m times of annealing, let $\hat{p}_{0rm}$ be DW success frequency out of m repetitions and $\hat{q}_{\ell rm}$, $\ell = 1, 2, 3$, the success frequencies for SA, SQA & SSSV

$H_{0r} : p_{0r\infty} = q_{\ell r\infty}$ vs $H_{ar} : p_{0r\infty} \neq q_{\ell r\infty}$

$$T_{r\ell} = \frac{m(\hat{p}_r - \hat{q}_{\ell,r})^2}{\hat{p}_r(1 - \hat{p}_r) + \hat{q}_{\ell,r}(1 - \hat{q}_{\ell,r})}$$

$$T^*_{r\ell} = 2m\left[\arcsin\left(\sqrt{\hat{p}_r}\right) - \arcsin\left(\sqrt{\hat{q}_{\ell,r}}\right)\right]^2$$

Multiple Statistical Tests

For the r-th instance, repeat m times of annealing, let $\hat{p}_{0rm}$ be DW success frequency out of m repetitions and $\hat{q}_{\ell rm}$, $\ell = 1, 2, 3$, the success frequencies for SA, SQA & SSSV

$H_{0r} : p_{0r\infty} = q_{\ell r\infty}$ vs $H_{ar} : p_{0r\infty} \neq q_{\ell r\infty}$

$$T_{r\ell} = \frac{m(\hat{p}_r - \hat{q}_{\ell,r})^2}{\hat{p}_r(1-\hat{p}_r) + \hat{q}_{\ell,r}(1-\hat{q}_{\ell,r})}$$

$$T^*_{r\ell} = 2m\left[\arcsin\left(\sqrt{\hat{p}_r}\right) - \arcsin\left(\sqrt{\hat{q}_{\ell,r}}\right)\right]^2$$

Asymptotic distribution under H_{0r}

As $m, n \to \infty$, if $\log n/m \to 0$, then

$$T_{r\ell} \longrightarrow \chi^2_1, \quad T^*_{r\ell} \longrightarrow \chi^2_1 \quad \text{uniformly over } r = 1, \cdots, n$$

Multiple Statistical Tests

For the r-th instance, repeat m times of annealing, let $\hat{p}_{0rm}$ be DW success frequency out of m repetitions and $\hat{q}_{\ell rm}$, $\ell = 1, 2, 3$, the success frequencies for SA, SQA & SSSV

$H_{0r} : p_{0r\infty} = q_{\ell r\infty}$ vs $H_{ar} : p_{0r\infty} \neq q_{\ell r\infty}$

$$T_{r\ell} = \frac{m(\hat{p}_r - \hat{q}_{\ell,r})^2}{\hat{p}_r(1-\hat{p}_r) + \hat{q}_{\ell,r}(1-\hat{q}_{\ell,r})}$$

$$T^*_{r\ell} = 2m\left[\arcsin\left(\sqrt{\hat{p}_r}\right) - \arcsin\left(\sqrt{\hat{q}_{\ell,r}}\right)\right]^2$$

Asymptotic distribution under H_{0r}

As $m, n \to \infty$, if $\log n/m \to 0$, then

$$T_{r\ell} \longrightarrow \chi^2_1, \quad T^*_{r\ell} \longrightarrow \chi^2_1 \quad \text{uniformly over } r = 1, \cdots, n$$

p-values & FDR

H_{0r} vs H_{ar} : p-value = $P(\chi^2_1 \geq T_{r\ell})$ p-value = $P(\chi^2_1 \geq T^*_{r\ell})$

Goodness-of-fit test

$H_0 : p_{0r\infty} = q_{\ell r\infty}$ for all $1 \leq r \leq n$ vs $H_a : p_{0r\infty} \neq q_{\ell r\infty}$ for some r

$$U_\ell = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n) \quad U_\ell^* = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n)$$

Goodness-of-fit test

$H_0 : p_{0r\infty} = q_{\ell r\infty}$ for all $1 \leq r \leq n$ vs $H_a : p_{0r\infty} \neq q_{\ell r\infty}$ for some r

$$U_\ell = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n) \quad U_\ell^* = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n)$$

Asymptotic distribution under H_0 as $m, n \to \infty$

$$U_\ell \to N(0,1) \quad U_\ell^* \to N(0,1)$$

Conditions

(1) $\sqrt{n}/m \to 0$.

(2) $p_{0r\infty} = q_{\ell r\infty}$=true success probability for method ℓ with the r-th instance are bounded away from 0 and 1.

Goodness-of-fit test

$H_0 : p_{0r\infty} = q_{\ell r\infty}$ for all $1 \le r \le n$ vs $H_a : p_{0r\infty} \neq q_{\ell r\infty}$ for some r

$$U_\ell = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n) \quad U_\ell^* = (2n)^{-1/2} \sum_{r=1}^{n} (T_{r\ell} - n)$$

Asymptotic distribution under H_0 as $m, n \to \infty$

$$U_\ell \to N(0,1) \quad U_\ell^* \to N(0,1)$$

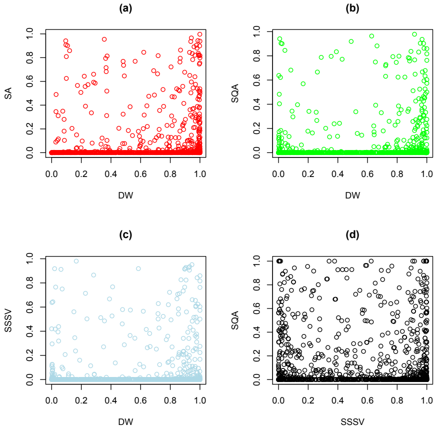
p-value = $2[1 - \Phi(|U_\ell|)]$ p-value = $2[1 - \Phi(|U_\ell^*|)]$

Conditions

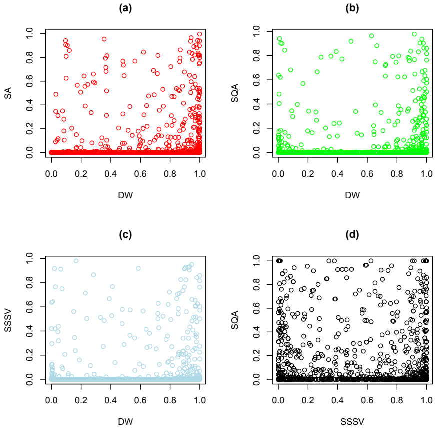
(1) $\sqrt{n}/m \to 0$.

(2) $p_{0r\infty} = q_{\ell r\infty}$=true success probability for method ℓ with the r-th instance are bounded away from 0 and 1.

Multiple Tests: FDR

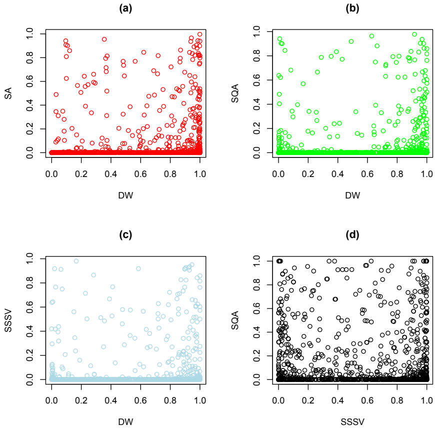


Multiple Tests: FDR



p-values

Multiple Tests: FDR



p-values

FDR

q-value = essentially zero

Goodness-of-fit-test

Goodness-of-fit-test

SQA vs DW

p-values = 0

Goodness-of-fit-test

SQA vs DW

p-values = 0

SSSV vs DW

p-values = 0

Goodness-of-fit-test

SA vs DW

p-values = 0

SQA vs DW

p-values = 0

SSSV vs DW

p-values = 0

Goodness-of-fit-test

Reject null hypothesis

all p-values $\leq 3.87 \times 10^{-6}$

SA vs DW

p-values = 0

SQA vs DW

p-values = 0

SSSV vs DW

p-values = 0

Goodness-of-fit-test

Reject null hypothesis

all p-values $\leq 3.87 \times 10^{-6}$

SA vs DW

p-values = 0

SQA vs DW

p-values = 0

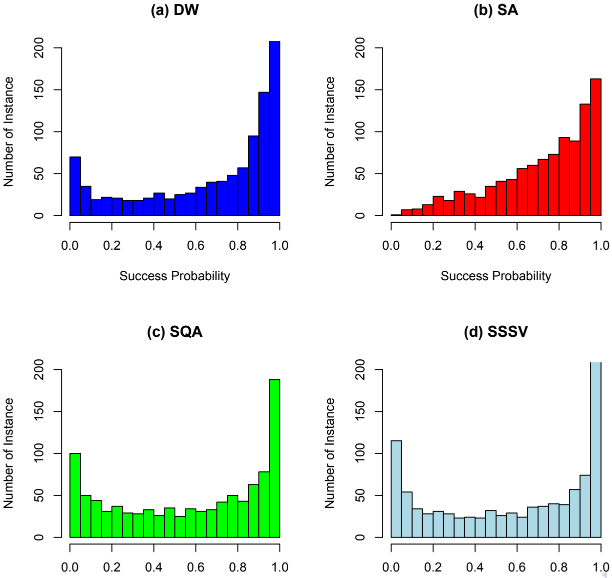
SSSV vs DW

p-values = 0

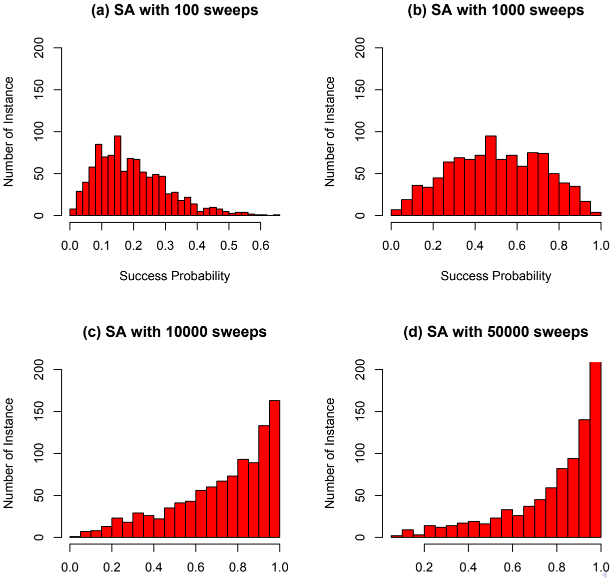
Conclusion: Overwhelming rejection

Overwhelming evidence to reject that DW is statistically consistent with SQA or SSSV in terms of ground state success probability

Histogram of Ground State Success Probability Data

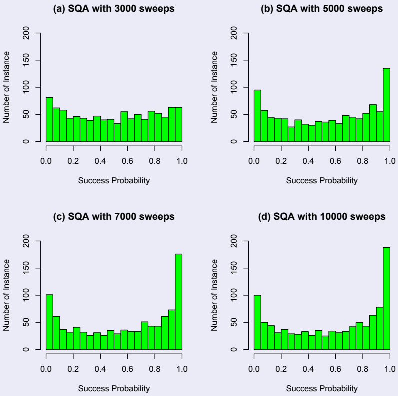


SA Histograms for different annealing times



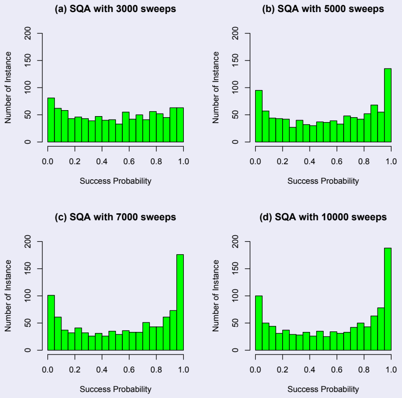
SQA Histograms

Various annealing times

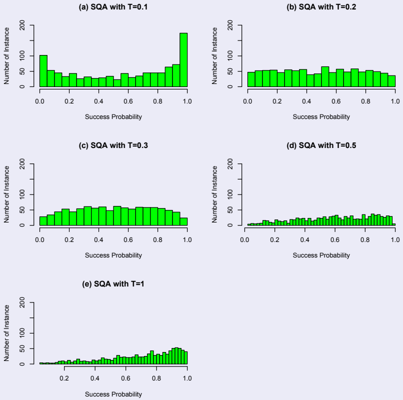


SQA Histograms

Various annealing times

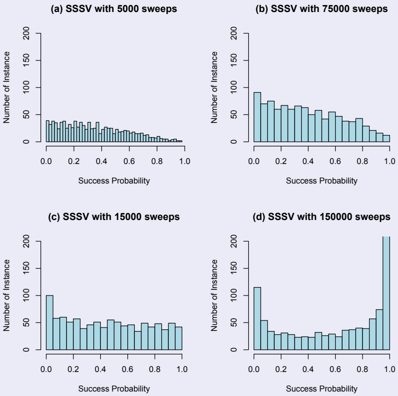


Various temperatures



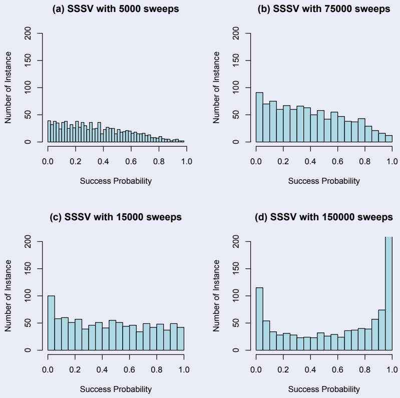
SSSV Histograms

Various annealing times

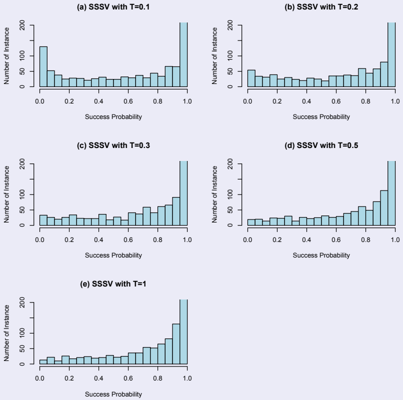


SSSV Histograms

Various annealing times



Various temperatures



DIP Test for Shape Patterns

$$DIP(F_n) = \max_{0 \le p \le 1} |F_n(p) - \hat{F}_n(p)|$$

F_n=empirical DF, $\hat{F}_n$= DF estimator under unimodality or U-shape

Under uniform null (asymptotic least favorable) distribution, as $n \to \infty$,

$$\sqrt{n} DIP(F_n) \to DIP(B), \quad B(t) = \text{ Brownian bridge on } [0, 1]$$

DIP Test for Shape Patterns

$$DIP(F_n) = \max_{0 \le p \le 1} |F_n(p) - \hat{F}_n(p)|$$

F_n=empirical DF, $\hat{F}_n$= DF estimator under unimodality or U-shape

Under uniform null (asymptotic least favorable) distribution, as $n \to \infty$,

$$\sqrt{n} DIP(F_n) \to DIP(B), \quad B(t) = \text{ Brownian bridge on } [0, 1]$$

Unimodality (including monotone)

DW: no

SA: yes

SQA: no

SSSV: no

DIP Test for Shape Patterns

$$DIP(F_n) = \max_{0 \le p \le 1} |F_n(p) - \hat{F}_n(p)|$$

F_n=empirical DF, $\hat{F}_n$= DF estimator under unimodality or U-shape

Under uniform null (asymptotic least favorable) distribution, as $n \to \infty$,

$$\sqrt{n} DIP(F_n) \to DIP(B), \quad B(t) = \text{ Brownian bridge on } [0, 1]$$

Unimodality (including monotone)

DW: no

SA: yes

SQA: no

SSSV: no

U-shape

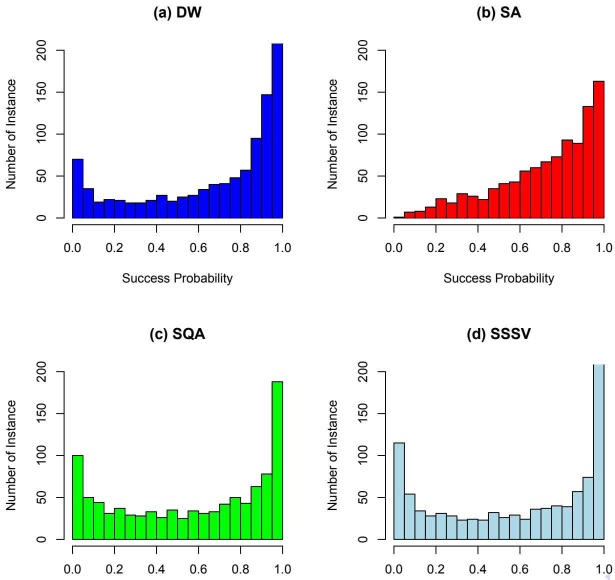
DW: no

SA: no

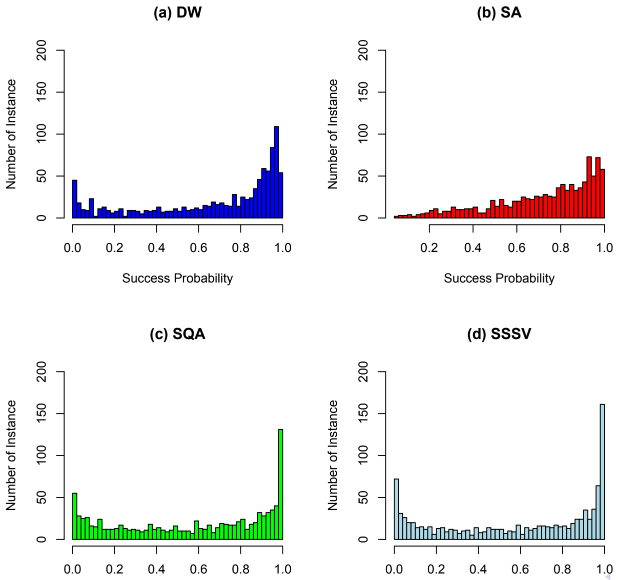
SQA: yes

SSSV: yes

Histogram of Success Probability



Histogram of Success Probability



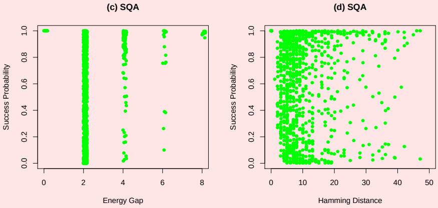
Shape Pattern Analysis by Regression

Covariates

Energy gap & Hamming distance between ground state and 1st excited state

Shape Pattern Analysis by Regression

SQA

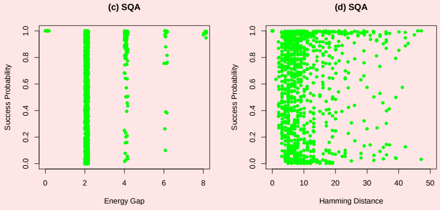


Covariates

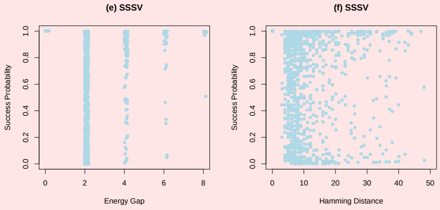
Energy gap & Hamming distance between ground state and 1st excited state

Shape Pattern Analysis by Regression

SQA



SSSV

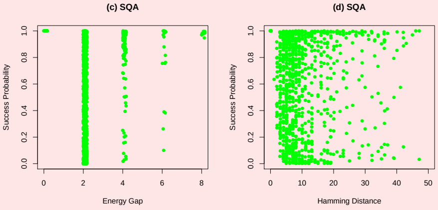


Covariates

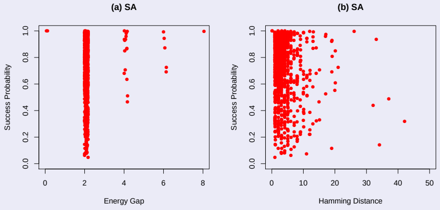
Energy gap & Hamming distance between ground state and 1st excited state

Shape Pattern Analysis by Regression

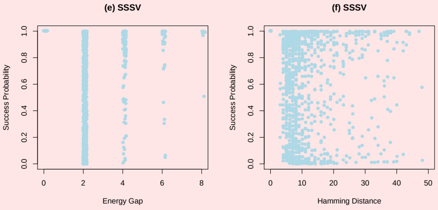
SQA



SA



SSSV

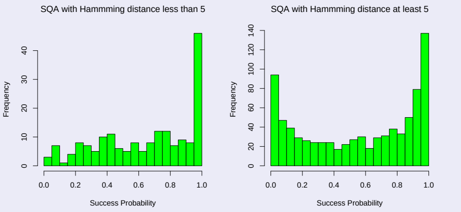


Covariates

Energy gap & Hamming distance between ground state and 1st excited state

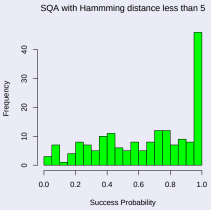
Shape Pattern Analysis by Regression

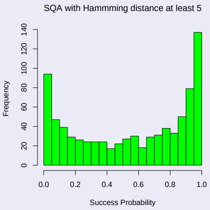
SQA



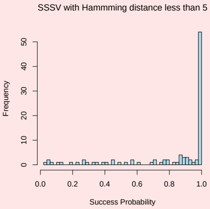
Shape Pattern Analysis by Regression

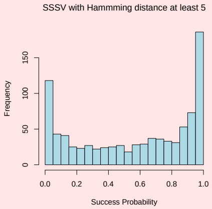
SQA





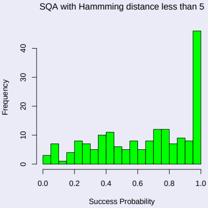
SSSV

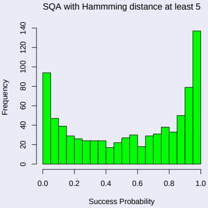




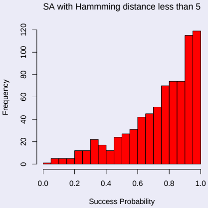
Shape Pattern Analysis by Regression

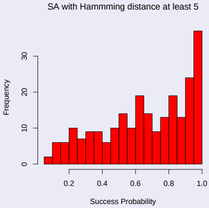
SQA



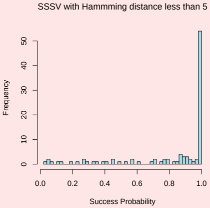


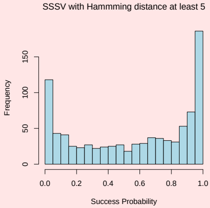
SA





SSSV





Concluding Remarks

Both inference and computing are inportant for big data.

Concluding Remarks

Both inference and computing are inportant for big data.

Interface

- Computing for conducting statistical inference; and statistics for analyzing computational algorithms.
- Statistics for quantum technology (e.g. quantum computing & tomography), and quantum computing for statistical computing and machine learning.